Linear Sorting

(☆ Motivation : Can sort in O(n) under certain conditions)

Better than comparison-based sorting algorithms

Sorting Algorithms

$N \stackrel{?}{=}$ # data items to sort

- Comparison Sorts
  - Bubble Sort $O(N^2)$
  - Heap Sort, Merge Sort $O(N \cdot \log N)$

- Stable vs unstable

9, 1, 3, 1, 4, 4, 3

(1) (2) (1) (2)

Counting Sort

R: 0, 1, 2, ..., 9

We take the range R of symbols into consideration.

Input sequence: 8, 8, 9, 0, 1, 3, 9, 0, 3, 5, 3

why going back to front: Guarantee the stability

index:	0	1	2	3	4	5	6	7	8	9
counts:	~~0~~ ~~1~~ 2	~~0~~ 1	0	~~0~~ ~~1~~ ~~2~~ 3	0	~~0~~ 1	0	0	~~0~~ ~~1~~ 2	~~0~~ ~~1~~ 2

index:	0	1	2	3	4	5	6	7	8	9
endpos:	~~1~~ ~~0~~ -1	~~2~~ 1	2	~~5~~ ~~4~~ ~~3~~ 2	5	~~6~~ 5	6	6	~~8~~ ~~7~~ 6	~~10~~ ~~9~~ 8

index:	0	1	2	3	4	5	6	7	8	9	10
output:	0	0	1	3	3	3	5	8	8	9	9

☆ We now start from the end of the input sequence to the start, and assign each value from the end-pos array

What if the range is big?

432, 534, 311, 119, 650, 903, 121, 777

$\Rightarrow$ Apply Counting Sort once per positions (3x)

$\rightarrow$ Radix Sort

Radix Sort, Iteration 1

(From least-significant to the most significant)

432, 534, 311, 119, 650, 903, 121, 777

index:	0	1	2	3	4	5	6	7	8	9
counts:	~~0~~ 1	~~0~~ ~~1~~ 2	~~0~~ 1	~~0~~ 1	~~0~~ 1	0	0	~~0~~ 1	0	~~0~~ 1
endpos:	~~0~~ -1	~~2~~ ~~1~~ 0	~~3~~ 2	~~4~~ 3	~~5~~ 4	5	5	~~6~~ 5	6	~~7~~ 6

output:

0	1	2	3	4	5	6	7
650	311	121	432	903	534	777	119

Radix Sort, Iteration 2

650, 311, 121, 432, 903, 534, 777, 119

index:	0	1	2	3	4	5	6	7	8	9
counts:	1	2	1	2	0	1	0	1	0	0
endpos:	-1	0	2	3	5	5	6	6	7	7

output:

0	1	2	3	4	5	6	7
903	311	119	121	432	534	650	777

Radix Sort, Iteration 3

903, 311, 119, 121, 432, 534, 650, 777

index:	0	1	2	3	4	5	6	7	8	9
counts:	0	2	0	1	1	1	1	1	0	1
endpos:	-1	-1	1	1	2	3	4	5	6	7

output:

0	1	2	3	4	5	6	7
119	121	311	432	534	650	777	903

Complexity

N: # of data items to sort

R: range of symbols

Q: length of the data items

Counting Sort: $O(N)$

Radix Sort: $O(N)$